Alfredo's Shop of Trinkets

FPGAs, ARM, microcontrollers, MSP430, mbed and image processing with a high dose of Touhou.

Tutorial: Switching clock modes with the FRDM KL25z in mbed

Posted on **September 2, 2014**

Ever since I got a hold of my FRDM-KL25Z board I fell in love with it. It has the convience of running straight from either the built in SDA USB connector, an external 5-9 volts supply or a regulated 3.3v supply. It boasts USB host, ultra low power modes, and when paired with the mbed online compilation polatform it essentially erased all my worries about working in several computers and file revisions. Once you use the mercurial based repository you will wonder why you ever did any project differently before. And it costs around 15.00 USD, so its a hard price break to beat for homegrown projects.

That being said, there are several disadvantages. Since you have to rely on the mbed library for mostly everything, some rather crucial functions are still not implemented, or probably fall outside the scope of the core mbed library. One of such functions is clock control. On the KL25z there is simply no way of switching the clock mode either than manually coding it. In this tutorial I will go through the steps of controlling the Multipurpose Clock Generator (MCG), or you can just skip the tutorial and get the library:

http://mbed.org/users/AlfredoER/code/KL25Z_ClockControl/

Operation modes

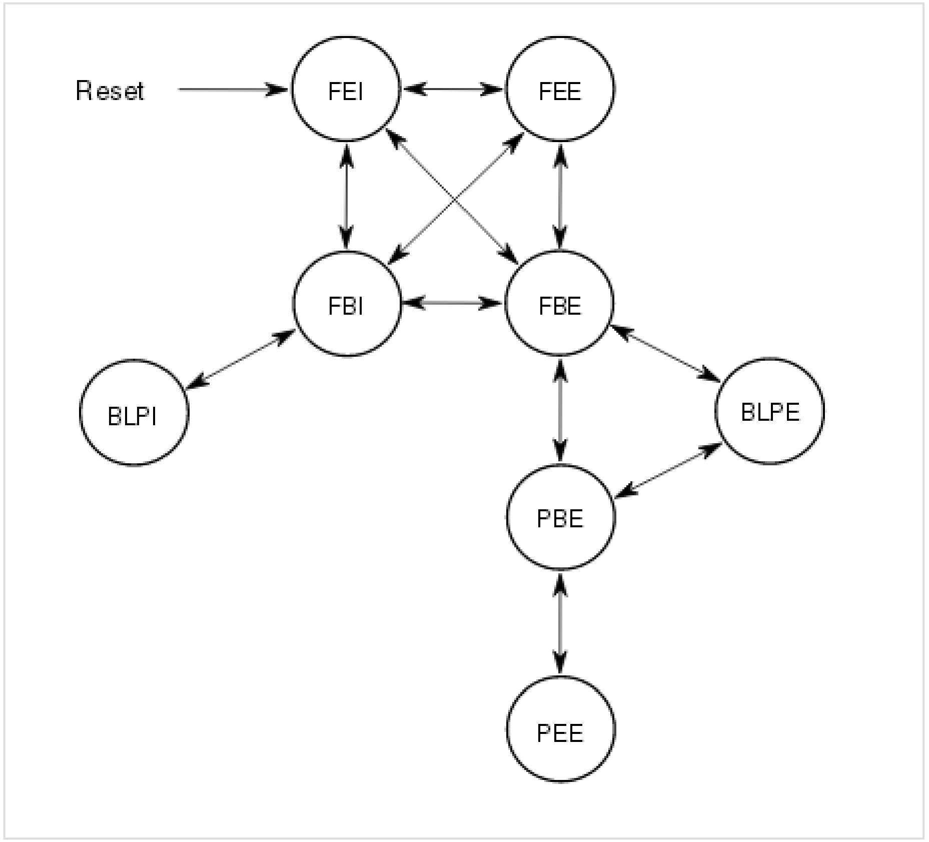
Starting off, there are several clock modes in the KL25z.

- FLL Engaged Internal (FEI) clock mode
- FLL Bypassed Internal (FBI) clock mode

- FLL Bypassed External (FBE) clock mode
- FLL Engaged External (FEE) clock mode
- PLL Bypassed External (PBE) clock mode
- PLL Engaged External (PEE) clock mode
- Bypassed Low Power External (BLPE) clock mode
- Bypassed Low Power Internal (BLPI) clock mode

The names are almost self explanatory, though it's worth mentioning some special aspects of some of them. FEI is the mode that the microcontroller starts in after a Power On Reset (POR). So any mode that you want to work in has to start from here. This mode sources the low frequency internal RC oscillator and pumps it up with the FLL to source the microcontroller. PEE is the mode that you probably want to use then running your application. It is the mode that sources an external high frequency cristal oscillator and pumps it up using the PLL for maximum speed and jitter free operation. This is the mode that consumes the most power but yields the maximum performance as well. BLPI is the mode that consumes the least energy and is excellent for low power modes, but is excruciatingly slow for any data transfers or operations in general.

The transitions between operating modes have to follow the state diagram shown below.



This tutorial will focus on the transitions between FEI, PEE and BLPI. The library that I’m publishing is still under work, and will eventually support transitions between all modes, but for the moment I’ll explain the steps for moving between those three modes. The reason is that they cover the basic clock requirements for most applications. FEI is a medium power mode that requires no external parts and is good for most applications, PEE will most likely be the mode that you use to get the most juice of the microcontroller and support high speed transfers to buses and is also necessary for running the USB Host with jitter free operation while BLPI will be the mode needed for low power operation and extending battery life. So, lets get started.

FEI to PEE

I’m assuming that you are using the FRDM KL25z board that comes with a stock 8MHz crystal. We will switch from the internal clock running on the FLL to a PLL derived clock that multiplies it up to 48 MHz.

- FEI to FBE

We will first make a transition to to the FLL Bypassed External (FBE) mode. This is not a mode that you actually want your appplication to run in. It is a mode that turns on the FLL and sets it up although the system clock will be directly driven from the external oscillator. This allows the FLL to acquire its target frequency without disrupting the system clock.

First off, enable the external clock oscillator by writing to C2.

```
/* MCG->C2: LOCRE0=0,??=0,RANGE0=2,HGO0=0,EREFS0=1,LP=0,IRCS=0 */
MCG->C2 = (uint8_t)0x24U;
```

Next, clear the C1[IREFS] bit to move to the external reference clock and set C1[CLKS] as 2'b10 to select the external clock as the system clock. C1[FRDIV] should also be set to divide the input to the FLL between 31.25 and 39.0625 kHz.

```
/* MCG->C1: CLKS=2,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */
MCG->C1 = (uint8_t)0x9AU;
```

Next, wait for the FLL reference to be the external clock and for the system clock to be soruced from the external clock as well.

```
while((MCG->S & MCG_S_IREFST_MASK) != 0x00U) ;
while((MCG->S & 0x0CU) != 0x08U) ;
```

- FBE to PBE

Afterwards, a transition to the PLL Bypassed External (PBE) mode is required. This too is not a real mode that you want to work in, but rather a stepping stone that turns on the PLL and stabilizes its output without actually running from it. (It also turns off the FLL on the way there). In my opinion, it should have been possible to switch directly to the PBE mode from FEI, but the state machine would have been even more complex, so I guess we can live with that.

Set C6[PLLS] to 1 to select the PLL and C6[VDIV0] to 1 so that the reference is multiplied by 12 and we end up with a system clock of 48 MHz.

```
MCG->C6 = (uint8_t)0x40U;
```

Now, wait for the external clock to be selected as the system clock and for the PLL to lock.

```
while((MCG->S & 0x0CU) != 0x08U);
while((MCG->S & MCG_S_LOCK0_MASK) == 0x00U) ;
```

- PBE to PEE

This is just the final step of the path. After having turned on the PLL and stabilized it to the frequency that we want we just switch the system clock to use the output of the PLL and presto!

```
MCG->C1 = (uint8_t)0x1AU;
```

Don't forget to wait for the output of the PLL to be selected as the source of the system clock.

```
while((MCG->S & 0x0CU) != 0x0CU) ;
```

PEE to BLPI

We have to go back the ladder here, and even lower into the basement so we can disable the external clock, disable the PLL, disable the FLL and only run the internal clock directly in low power mode. Lets get started:

- PEE to PBE

This is done by switching the system clock to the external source directly.

```
MCG->C1 = (uint8_t)0x90U;
```

Then we wait for the selection to be reflected by the system.

```
while((MCG->S & MCG_S_CLKST_MASK) != 0x2U<<MCG_S_CLKST_SHIFT);
```

- PBE to FBE

This is simply a move that selects the FLL while still sourcing the clock on the external oscillator.

```
MCG->C6 = (uint8_t)0x00U;
```

As usual, wait for the config to set in.

while((MCG->S & MCG_S_PLLST_MASK);

- FBE to FBI

This is the part where we switch to the internal clock.

MCG->C1 = (uint8_t)0x54U;

And wait for the internal clock is selected for the FLL and the internal reference feeds the system clock.

while((MCG->S & MCG_S_IREFST_MASK);

while((MCG->S & MCG_S_CLKST_MASK);

- FBI to BLPI

This final step only involves turning off the FLL. (Even though we never used it…) Just set the LP bit to get the FLL to fall asleep.

MCG->C2 = (uint8_t)0x02;   /* LP is 1 */

Anyway, we have reached our goal of going into the lowest power clock mode available for the KL25z. 😀

BLPI to FEI

This is a little shortcut. Instead of writing the whole code for transitioning into PEE again, I though we could just go back to FEI and use the code we already wrote to go into PEE again. This is used to exit low power mode and return to work at normal speed.

- BLPI to FBI

This is a simple step. Just drop the low power mode to get the FLL running again.

MCG->C2 = (uint8_t) 0x00;   /* LP is 0 */

- FBI to FEI

This step involves selecting the output of the FLL as the system clock.

```
MCG->C1 = (uint8_t)0x06U;
```

And waiting for the output to reflect the change.

```
while((MCG->S & 0x0CU) != 0x00U) ;
```

And that's it! We have successfully gone from start-up, to high power PLL engaged PEE mode, to low power BLPI mode and back again. I have implemented and tested the routines in the FRDM KL25z and can confirm the clock frequency changes working correctly. Happy switching!

This entry was posted in **Microcontrollers** and tagged **Clock Modes**, **FRDM**, **Freedom**, **KL25z**, **Low power**, **MCG** by **AlfredoER**. Bookmark the **permalink [http://alfredoer.com/microcontrollers-2/tutorial-switching-clock-modes-with-the-frdm-kl25z-in-mbed/]** .

4 THOUGHTS ON "TUTORIAL: SWITCHING CLOCK MODES WITH THE FRDM KL25Z IN MBED"

Hugo Perez
on **September 5, 2014 at 1:13 am** said:

Hi, thank you very much for the tutorial but mainly for sharing the library with the world, i will use it in my project if you don't mind.

LLUIS
on **October 2, 2014 at 9:36 pm** said:

debugging with P&E multilink it lost connection and terminates debug sessions at step "MCG->C1 = (uint8_t)0x54U;" anybody knows why this happen???

Diego
on **November 29, 2014 at 12:18 am** said:

Another useful tutorial.
I posted the same question on forum
In a KL05 project,recovering from VLLS3 and using a 32768 KHz crystal(always on in VLLS3) in FEE mode,more than 100 milliseconds are taken waiting IREFST bit goes zero.
Where my wrong setting could be?How can this be avoided?
Thanks Erich

alfredoer
on **December 9, 2014 at 2:45 am** said:

Hi Eric!
I haven't gotten around too much to playing the the low leakage stop modes, but if I recall correctly, the normal recovery method for VLLS3 is a wake up reset. http://cache.freescale.com/files/32bit/doc/prod_brief/KL0xPB.pdf pg 21. Since it is a stop mode, the clocks are stopped and the reset should clear the MCG to its default state, after which you should be able to switch to FEE and the loop should stabilize in a few hundred microseconds at the most. Have you tried a VLPS mode instead to test where the problem could be?